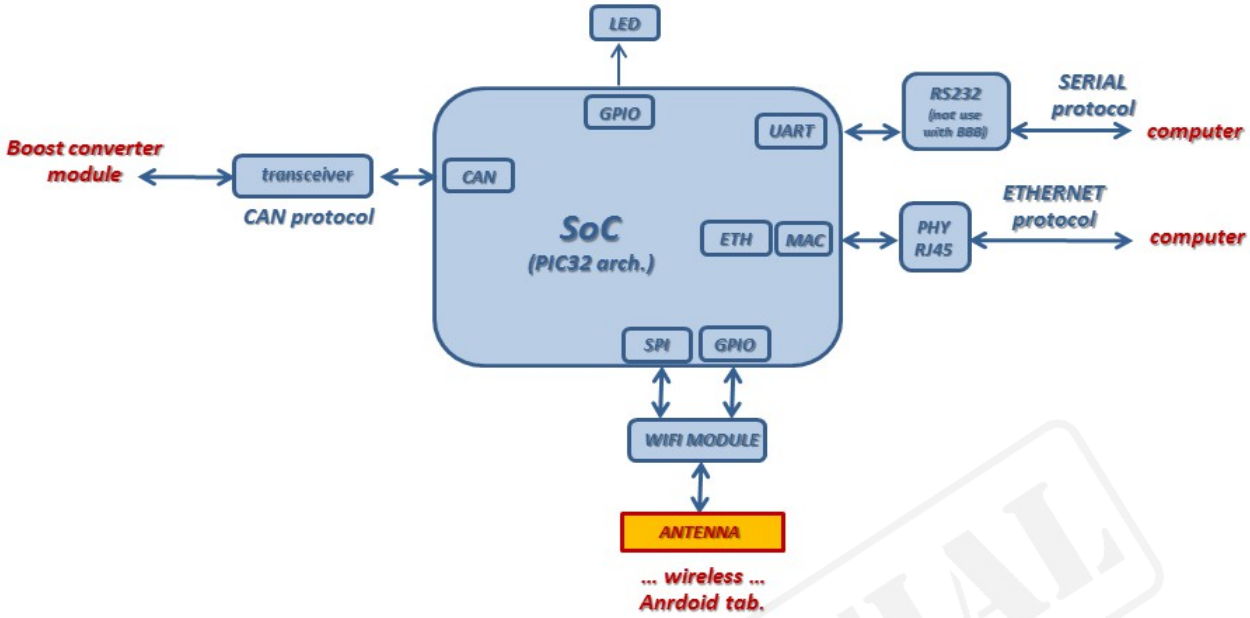
5.2. Interfaces

Intéressons nous aux interfaces de communications de l'application. Observons l'architecture matérielle de l'application (pour MCU PIC32MX) :



- ***Interface CAN :*** *l'interface CAN a été présentée plus en détail dans la partie BUS CAN de ce même document.*

- ***Interface LED :*** *L'application devra détecter la présence ou non d'un module boost (convertisseur + MCU) connecté. En l'absence de module la LED devra clignoter avec une périodicité de 500ms (500ms sec allumée, 500ms sec éteinte …). En cas de présence la LED restera allumée.*

- ***Interface RESEAU IP-ETHERNET (RJ45) :*** *l'interface ethernet RJ45 proposera un serveur TCP (ou UDP ou les deux) embarqué côté SoC. Toutes les trames échangées se feront via échanges de chaînes de caractères (aucun format entier ou flottant).* ***Côté ordinateur, si vous n'avez pas le temps de proposer une interface graphique de communication (interface GTK, QT ...), votre application devra par défaut proposer la même interface que le module UART (cf. shell). L'interface sera par exemple accessible depuis un client TCP côté ordinateur (par exemple Telnet).*** *Pour la partie ethernet RJ45, nous travaillerons en IP fixe reconfigurable par liaison série.*

- ***Interface RESEAU IP-WIFI :*** *l'interface WIFI proposera un serveur TCP (ou UDP ou les deux) embarqué côté SoC. Toutes les trames échangées se feront via échanges de chaînes de caractères (aucun format entier ou flottant). Une présentation plus détaillée de la partie Android et interface utilisateur sera faîte par la suite (partie INTERFACE UTILISATEUR).* ***Si possible, deux mode de fonctionnement doivent être supportés. Le mode AD HOC (imposé, doit fonctionner) et si possible le mode infrastructure ou Acces point (sans SSID).*** *Dans tous les cas, toutes connexions doit se faire vers le host sans identification. Le mode de fonctionnement (ad hoc ou infrastructure) sera géré à la compilation et ne sera pas modifiable en cours de fonctionnement.*